details of the actual programming associated with these objects in large part are not represented in the OED and are not crucial to an understanding of the present invention. The actual programming can be described in documentation separate from the OED, as discussed in more detail below.

[0054] The OEDs of the present invention have many uses. They may be used to document pre-existing software. Alternately, they may be used during software development as a means for a business logic software developer (architect) to describe the business logic of a GUI application at a high level without defining the appearance of the actual GUI programming elements, which details are left to the GUI programmer who actually generates the code. Accordingly, in one use, the web page **200** (comprising windows **201***a* and **201***b*) can pre-exist the OEDs and OEDs can be created for windows **201***a* and **201***b* from the web page (or, for that matter, from the code that generates the windows). Alternately, the OEDs can be created by an application architect and given to a programmer who will write the code for generating the windows based on the OEDs. This frees the business logic architect from having to be concerned with the presentation logic. Likewise, it provides the GUI programmer with all the information about the business logic necessary to build the GUI, but gives him total freedom to develop the "look and feel" of the interface. Thus, both the business logic architect and the GUI programmer can concentrate on the area of their specific expertise without being concerned about the other aspects of the application program. **FIG. 5** is an OED in accordance with the present invention diagram corresponding to the FLIGHTS window **201***a* shown in **FIGS. 2 and 3** and **FIG. 6** is an OED diagram in accordance with the present invention corresponding to the FLIGHTS RESULTS window **201***b* shown in **FIGS. 2 and 4**.

[0055] Before describing the OEDs shown in **FIGS. 5 and 6**, a description of at least one particular exemplary set of rules and recommendations for preparing OEDs in accordance with one embodiment of the invention using the symbol library shown in **FIG. 1** is appropriate.

[0056] Each OED should have a particular main object. For any given application, any object sufficiently complex to require specific description in order to properly to enable a programmer to write the desired code or sufficiently complex to warrant separate documentation after written using traditional criteria and common sense should have its own OED. The main object of an OED may be almost any object type, but most often will be higher level dynamic object such as a window, as illustrated in each of **FIGS. 5 and 6**, or an overall system, application or project (i.e., an application object), as illustrated in **FIG. 8** to be discussed further below. Since such objects tend to be complicated, i.e., have many other objects associated therewith or defined there within. On the other hand, static objects, such as buttons and data structures, and lower level dynamic objects, such as scripts often are quite simple and/or self-explanatory so that they do not require a separate OED.

[0057] In a preferred embodiment of the invention, the OEDs are never used to represent program calls. Thus, linking a method object to a window object in an OED does not mean that the window will call that method. Rather, it means that the method is available in that window and any event script assigned to the window or to another object within the window object can invoke that method.

[0058] The preparer of the OED (let us assume it is a program architect preparing an OED for use by a programmer in generating GUI code) should start an OED by drawing (or dragging and dropping in the case of a software implementation of the invention) the main object of the particular OED, whether it is an application, a window or something else. The architect should then place a circle around it. The object with the circle around it will be called the main object and it is the object that is being defined in that particular OED. In a preferred embodiment, any inheritance of the main object is represented within the big circle.

[0059] Referring to **FIG. 5**, for example, it shows an OED **501***a* describing the main FLIGHTS window **201***a* shown in **FIG. 3** of the web page **200** of **FIG. 2**. As noted above, this exact window may be used in a number of the pages of the web site. The main flights window OED **500***a* is developed by first drawing a window object symbol **501**. Next, a big circle **503** is drawn around window symbol **501** to define it as the main object of OED **500**. Preferably, inheritance is represented within the big circle **503**. Accordingly, inheritance symbol **505** and the class that is the source of the inherited characteristics is drawn within the circle **503**. In this case, the source is the class "windows", and it is represented by a class symbol **507**. The inheritance symbol **505** should be drawn between the object inheriting the features, namely, window object **501**, and the object from which it is inheriting those features, namely, class object **507**, with the arrow pointing toward the object that is inheriting the features. Event scripts that are executed upon the opening or closing of the main object, i.e., the FLIGHTS window **501**, also may be placed within the big circle **503**. In this particular example, there is a script for each of those events and they are represented in the drawing by script symbols **509** and **511** corresponding to scripts executed upon opening and closing, respectively, of the FLIGHTS window **501**. The SCRIPTS **509** and **511** are connected by a simple line to the object to which they are assigned. Event script symbols such as symbols **509** and **511** in **FIG. 5** do not represent the code module, e.g., method, that is invoked by the script. They represent the script. The method invoked by the event script is separately represented in an OED with a method type symbol. The OED or collection of OEDs that represent an application should show that the method invoked by the event script is available to the main object of the OED on which the event script appears. Thus, as will be seen from the discussion further below, the relevant method should be shown either in this OED linked to window symbol **501** or, alternatively, in the OED of another object from which the main object, window **501**, of this OED has inheritance.

[0060] In a preferred embodiment of the invention, essentially everything else is shown outside of the big circle, including event scripts that are executed responsive to any events other than opening and closing of the main object of the diagram. Except for objects that are logically connected to another object by one of the three previously defined relationships that have their own symbols (namely, inheritance, data transfer, and remote link), all other relationships between objects in an OED (i.e., being assigned to another object or being defined within another object) preferably are represented by a simple line between the two objects. Generally, the relationship between the objects so linked will be self-explanatory simply based on the types of the two objects. Thus, the lines normally would not need an arrow at